

Consensus-Based Modelling using Distributed Feature Construction^{*}

Haimonti Dutta¹ and Ashwin Srinivasan²

¹ Department of Management Science and Systems
University at Buffalo
New York, NY 14260.
haimonti@buffalo.edu

² Department of Computer Science
IIT, Delhi.
ashwin@iiitd.ac.in

Abstract. A particularly successful role for Inductive Logic Programming (ILP) is as a tool for discovering useful relational features for subsequent use in a predictive model. Conceptually, the case for using ILP to construct relational features rests on treating these features as functions, the automated discovery of which necessarily requires some form of first-order learning. Practically, there are now several reports in the literature that suggest that augmenting any existing features with ILP-discovered relational features can substantially improve the predictive power of a model. While the approach is straightforward enough, much still needs to be done to scale it up to explore more fully the space of possible features that can be constructed by an ILP system. This is in principle, infinite and in practice, extremely large. Applications have been confined to heuristic or random selections from this space. In this paper, we address this computational difficulty by allowing features to be constructed in a distributed manner. That is, there is a network of computational units, each of which employs an ILP engine to construct some small number of features and then builds a (local) model. We then employ a consensus-based algorithm, in which neighbouring nodes share information to update local models. For a category of models (those with convex loss functions), it can be shown that the algorithm will result in all nodes converging to a consensus model. In practice, it may be slow to achieve this convergence. Nevertheless, our results on synthetic and real datasets that suggests that in relatively short time the “best” node in the network reaches a model whose predictive accuracy is comparable to that obtained using more computational effort in a non-distributed setting (the best node is identified as the one whose weights converge first).

^{*} A short 6-page version of this paper was presented at the 24th International Conference on Inductive Logic Programming, held in conjunction with ECML PKDD, France. Furthermore, a significant part of the work in this paper was done when the first author was an Associate Research Scientist at the Center for Computational Learning Systems (CCLS), Columbia University, NY.

1 Introduction

The field of Inductive Logic Programming (ILP) has made steady progress over the past two decades, in advancing the theory, implementation and application of logic-based relational learning. A characteristic of this form of machine-learning is that data, prior knowledge and hypotheses are usually—but not always—expressed in a subset of first-order logic, namely logic programs. Side-stepping for the moment the question “why logic programs?”, it is evident that settling on some variant of first-order logic allows the construction of tools that enable the automatic construction of descriptions that use relations (used here in the formal sense of a truth value assignment to n -tuples).

There is at least one kind of tasks where some form of relational learning would appear to be necessary. This is to do with the identification of functions (again used formally, in the sense of being a uniquely defined relation) whose domain is the set of instances in the data. An example is the construction of new “features” for data analysis based on existing relations (“ $f(m) = y$ if a molecule m has 3 or more benzene rings fused together otherwise $f(m) = n$ ”). Such features are not intended to constitute a stand-alone description of a system’s structure. Instead, their purpose is to enable different kinds of data analysis to be performed better. These may be constructing models for discrimination, joint probability distributions, forecasting, clustering, and so on. If a logic-based relational learner like an ILP engine is used to construct these relational features, then each feature is formulated as a logical formula. A measure of comprehensibility will be retained in the resulting models that use these features.

The approach usually, but not always, separates relational learning (to discover features) and modelling (to build models using these features). There will of course be problems that require the joint identification of relational with features and models—the emerging area of statistical relational learning (SRL), for example, deals with the conceptual and implementation issues that arise in the joint estimation of statistical parameters and relational models. It would appear that separate construction of features and statistical models would represent no more than a poor man’s SRL. Nevertheless, there is now a growing body of research that suggests that augmenting any existing features with ILP-constructed relational ones can substantially improve the predictive power of a statistical model (see, for example: [29, 49, 52, 47, 53]). There are thus very good practical reasons to persist with this variant of statistical and logical learning for data analysis.

There are known shortcomings with the approach which can limit its applicability. First, the set possible relational features is usually not finite. This has led to an emphasis on syntactic and semantic restrictions constraining the features to some finite set. In practice this set is still very large, and it is intractable to identify an optimal subset of features. ILP engines for feature-construction therefore employ some form of heuristic search. Second, much needs to be done to scale ILP-based feature discovery up to meet modern “big” data requirements. This includes the abilities to discover features using very large datasets not all stored in one place, and perhaps only in secondary memory; from rela-

tional data arriving in a streaming manner; and from data which do not conform to expected patterns (the concept changes, or the background knowledge becomes inappropriate). Third, even with “small” data, it is well-known that obtaining the value of a feature function for a data instance can be computationally hard. This means that obtaining the feature-vector representation using ILP-discovered features can take large amounts of time.

This paper is concerned only with the first of these problems, namely how to construct models when feature-spaces are very large. Each node has access to some (but not all) relational features constructed by an ILP engine, and constructs a local linear model. Using a simple consensus-based algorithm, all nodes in the network converge on the optimal weights for all the features. The setting is naturally amenable to distributed learning, providing us with a mechanism of scaling-up the construction of models using ILP-based feature discovery. Our approach does not result an optimal answer to the feature-selection problem. However it does have some positive aspects: it provides one way of distributing the computational task of feature-construction; and given multiple, possibly overlapping sets of relational features, it provides one way of identifying the best model (for a specific category of models). Figure 1 illustrates this. The algorithm for distributed feature estimation, described in Section 3.1 is an implementation of such an approach.

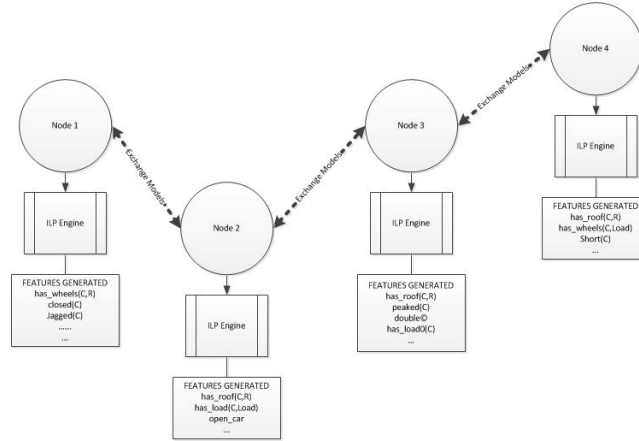


Fig. 1. An illustrative example of the consensus-based approach using Michalski’s “Trains” problem [35]. Each node has local features generated from an ILP engine, builds local models, estimates loss and shares information with its neighbours. Eventually we would like all nodes converge to the same model.

Section 2 presents related work; Section 3.1 formulates the problem as one of consensus-based model construction. Section 3.2 presents an iterative procedure for constructing models in a network of nodes capable of exchanging informa-

tion about their local models. Experimental results are in Section 4. Section 5 discussed open issues and concludes the paper.

2 Related Work

Techniques for selecting from a (large) but finite set of features of known size d has been well-studied within the machine learning, usually under the umbrella-terms of filter-based or wrapper-based methods (see for example, [28, 36]). While most of the early work was intended for implementation on a single machine, feature-selection from fixed-sized feature spaces has been extended to a distributed setting for large datasets. Here, the data are partitioned and placed on different processors; processors must communicate to find parameters that minimize loss over the entire dataset. However, communication has to be fast enough so that network latencies do not offset computational gains. The partitioning scheme – horizontal (all instances have the same features) or vertical (instances have access to only a subset of features) plays an important role in the design of these distributed algorithms. Early work in decentralized optimization was marked by interest in consensus-based learning, distributed optimization and minimization with the seminal work of Bertsekas, Tsitsiklis and colleagues ([60, 59, 7]). More recently, researchers have shown that convergence properties of these decentralized algorithms can be related to the network topology by using spectral properties of random walks or path averaging arguments on the underlying graph structure ([11, 51, 20, 6]). Learning feature subsets in distributed environments using decentralized optimization has become an active area of research ([21, 1, 15]) in recent years.

Agarwal et al. [1] present a system and a set of techniques for learning linear predictors with convex losses on terabyte sized datasets. Their goal is to learn problems of the form $\min_{w \in R^d} \sum_{i=1}^n l(w^T x_i; y_i) + \lambda R(w)$ where x_i is the feature vector of the i^{th} example, w is the weight vector and R is a regularizer. The data are split horizontally and examples are partitioned on different nodes of a cluster. Duchi et al. [21] present a dual averaging sub-gradient method which maintains and forms weighted averages of sub-gradients in the network. An interesting contribution of this work is the association of convergence of the algorithm with the underlying spectral properties of the network. Similar techniques for learning linear predictors have been presented elsewhere ([37, 38, 62], [42, 12]). The algorithm presented in this paper differs from this body of literature in that the data are split *vertically* amongst nodes in the cluster thereby necessitating a different algorithm design strategy. In addition, this is a batch algorithm and hence quite different from distributed online learning counterparts ([19, 34, 9]).

Das et al. [18] show that three popular feature selection criteria – misclassification gain, gini index and entropy can be learnt in a large peer-to-peer network. This is then combined with protocols for asynchronous distributed averaging and the secure sum protocols to present a privacy preserving asynchronous feature selection algorithm.

Existing literature on discovering a subset of interesting features from large, complex search spaces such as those by ILP engines adapt one of the following strategies:

1. Optimally [23, 43, 33] or heuristically [29, 49, 40, 14, 52, 47, 53] solve a discrete optimization problem.
2. Optimally [24, 41] solve a convex optimization problem with sparsity inducing regularizers;
3. Compute all relational features that satisfy some quality criterion by systematically and efficiently exploring a prescribed search space [44, 27, 4, 3, 45, 2, 46, 5, 22].

Again, much of this has been of a non-distributed nature, and usually assume a bound on the size of the feature-space. The latter is not the case for a technique like the one proposed in [29]. This describes a randomized local search based technique which repeatedly constructs features and then performs a greedy local search starting from this subset. Since enumeration of all local moves can be prohibitively large, the selection of moves is guided by errors made by the model constructed using the current set of features. Nothing is assumed about the size of the feature-space, making it a form of vertical partitioning of the kind we are interested in. Multiple random searches can clearly be conducted in parallel (although this is not done in the paper). As with most randomised techniques of this kind, not much can be said about the final model.

Perhaps of most interest to the work here is the Sparse Network Of Winnow classifiers described in [48, 13]. As it stands, this horizontally partitions the data into subsets, constructs multiple linear models using Winnow’s multiplicative update process, and finally uses a majority vote to arrive at a consensus classification. This would appear, on the surface to be quite different to what we propose here. Nevertheless, there are reasons to believe that this approach can be usefully extended to the setting we propose. It has been shown elsewhere that the Winnow-based approach can be extended to an infinite-attribute setting [?]. The work in this paper shows that consensus linear models are possible when convex cost functions are used. Finally, from the ILP-viewpoint, [55] shows how it is possible to construct Winnow-based models in an infinite-attribute setting using an ILP engine with a stream-based model of the data. Taken together, this suggests that a combination of the techniques we propose, and those in [13] can be used to develop linear models that can handle both horizontal partitioning of the data and vertical partitioning of the feature-space.

3 Consensus-Based Model Construction

3.1 Problem Description

Let M denote an $n \times m$ matrix with real-valued entries. This matrix represents a dataset of n tuples of the form $X_i \in \mathbb{R}^m, 1 \leq i \leq n$. Assume, without loss of generality, this dataset has been vertically distributed over k sites S_1, S_2, \dots, S_k

Input: $n \times m_i$ matrix at each site S_i , $G(V, E)$ which encapsulates the underlying communication framework, T : no of iterations

Output: Each site S_i has $W_i \approx W_g[1 : m_i]$

for $t = 1$ **to** T **do**

- (a) Site S_i computes $M_i W_i^T$ locally and estimates the loss function;
- (b) Site S_i gossips with its neighbors $S_j \in \{N_i\}$ and obtains $M_j W_j^T$ for each neighbor;
- (c) Site S_i locally updates its function estimate as $J_i^t = \alpha_{ii}(M_i W_i^T) + \alpha_{ji}(M_j W_j^T)$;
- (d) Update the local weight vectors using stochastic gradient descent as follows: $\frac{\partial L_p}{\partial W_i} = -X_p(Y_p - J_i^t(W_i^t(p)))$;
- (e) If there is no significant change in the local weight vectors of one of the sites then stop

end

Algorithm 1: Consensus-Based Modelling

i.e. site S_1 has m_1 features, S_2 has m_2 features and so on, such that $|m_1| + |m_2| + \dots + |m_k| = |m|$, where $|m_i|$ represents the number of features at site S_i ³. Let M_1 denote the $n \times m_1$ matrix representing the dataset held by S_1 , M_2 denote the $n \times m_2$ matrix representing the dataset held by S_2 and so on. Thus, $M = M_1 : M_2 : \dots : M_k$ denotes the concatenation of the local datasets.

We want to learn a linear discriminative function over the data set M . The global function to be estimated is represented by $f_g = MW_g^T$ where W_g is assumed to be a $1 \times m$ weight vector. If only the local data is used, at site S_1 , the local function estimated would be $f_1 = M_1 W_1^T$. At site S_2 , the local function estimated would be $f_2 = M_2 W_2^T$. The goal is to describe a de-centralized algorithm for computing the weight vectors at sites S_1, \dots, S_k such that on termination $W_1 \approx W_g[1 : m_1], W_2 \approx W_g[1 : m_2], \dots, W_k \approx W_g[1 : m_k]$ where $W_g[1 : m_i]$ represents the part of the global weight vector for the attributes stored at that site S_i . Clearly, if all the datasets are transferred to a central location, the global weight vector can be estimated. Our objective is to learn the function in the decentralized setting assuming that transfer of actual data tuples is expensive and may not be allowed (say for example due to privacy concerns). The weights obtained at each site on termination of the algorithm will be used for ranking the features.

3.2 Algorithm

We state the following assumptions under which the distributed algorithm operates:

Assumption 1: Model of Distributed Computation. The distributed algorithm evolves over discrete time with respect to a “global” clock⁴. Each site has

³ In the more general setting, Site S_i has a random subset of features $m_i \subset m$.

⁴ Existence of this clock is of interest only for theoretical analysis.

access to a local clock. Furthermore, each site has its own memory and can perform local computation (such as computing the gradient on its local attributes). It stores f_i , which is the estimated local function. Besides its own computation, sites may receive messages from their neighbors which will help in evaluation of the next estimate for the local function.

Assumption 2: Communication Protocols. Sites S_i are connected to one another via an underlying communication framework represented by a graph $G(V, E)$, such that each site $S_i \in \{S_1, S_2, \dots, S_k\}$ is a vertex and an edge $e_{ij} \in E$ connects sites S_i and S_j . Communication delays on the edges in the graph are assumed to be zero. It must be noted that the communication framework is usually expected to be application dependent. In cases where no intuitive framework exists, it may be possible to simply rely on the physical connectivity of the machines, for example, if the sites S_i are part of a large cluster.

Algorithm 1 describes how the weights for attributes will be estimated using a consensus-based protocol. There are two main sub-parts of the algorithm: (1) Exchange of local function estimate and (2) Local update based on stochastic gradient descent. Each of these sub-parts are discussed in further detail below. Furthermore, assume that $J : R^m \rightarrow [0, \infty]$ is a continuously differentiable non-negative cost function with a Lipschitz continuous derivative.

Exchange of Local Function Estimate: Each site locally computes the loss based on its attributes and then gossips with the neighbors to get information on other attributes. On receiving an update from a neighbor, the site re-evaluates J_i by forming a component-wise convex combination of its old vector and the values in the messages received from neighbors i.e. $J_i^{t+1} = \alpha_{ii}(X_i W_i^T) + \alpha_{ji}(X_j W_j^T)$. It is interesting to note that $\alpha_{il}, 0 \leq \alpha_{il} \leq 1$, is a non-negative weight that captures the fraction of information site i is willing to share with site l . The choice of α_{il} may be deterministic or randomized and may or may not depend on the time t [30]. The $k \times k$ matrix A comprising of $\alpha_{il}, 1 \leq i \leq k, 1 \leq l \leq k$ is a “stochastic” matrix such that it has non-negative entries and each row sums to one. More generally, this reflects the state transition probabilities between sites. Figure 2 illustrates the state transition between two sites S_i and S_j .

Another interpretation of the diffusion of J_i amongst the neighbors of i involves drawing analogies from Markov chains – the diffusion is mathematically identical to the evolution of state occupation probabilities. Furthermore, a simple vector equation can be written for updating J_i^t to J_i^{t+1} i.e. $J_i^{t+1} = A(i)(J_i^t)_{N_i}$ where $A(i)$ corresponds to the row i of the matrix A and $(J_i^t)_{N_i}$ is a matrix that has $|N_i|$ rows (each row corresponding to a neighbor of Site S_i) and n columns (each column corresponding to all the instances). More generally, $\mathcal{J}^{t+1} = A\mathcal{J}^t$ where \mathcal{J}^{t+1} is a $k \times n$ matrix storing the local function estimates of each of the n instances at site k and A is the $k \times k$ transition probability matrix corresponding to all the sites. It follows that $\lim_{t \rightarrow \infty} A^t$ exists and this controls the rate of convergence of the algorithm.

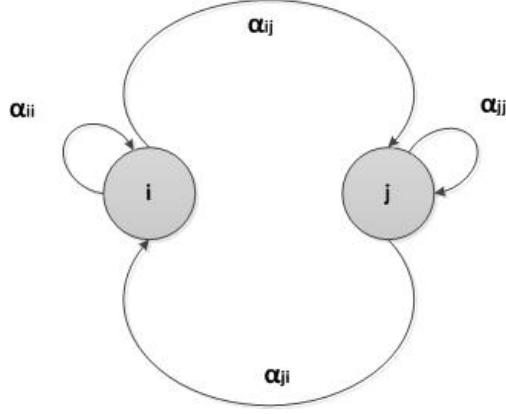


Fig. 2. State Transition Probability between two sites S_i and S_j

We introduce the notion of *average function estimate* in the network $\mathbf{J}_i^t = \sum_i \frac{J_i^t}{k}$ which allocates equal weight to all the local function estimates and serves as a baseline against which individual sites S_i 's can compare their performance. Philosophically, this also implies that each local site should at least try to attain as much information as required to converge to the average function estimate. Since $\sum_i \alpha_{ij} = 1$, this estimate is invariant.

The A matrix has interesting properties which allow us to show that convergence to \mathbf{J}_i^t occurs. One such property is the Perron-Frobenius theory of irreducible non-negative matrices. We state the theorem here for continuity.

Theorem 1 Perron-Frobenius [61] *Let A be a positive, irreducible matrix such that the rows sum to 1. Then the following are true:*

1. *The eigenvalues of A of unit magnitude are the k -th roots of unity for some k and are all simple.*
2. *The eigenvalues of A of unit magnitude are the k -th roots of unity if and only if A is similar under a permutation to a k cyclic matrix*
3. *all eigenvalues of A are bounded by 1.*

Since the eigenvalues of A are bounded by 1, it can be shown that J_i^t converges to the average function estimate \mathbf{J}_i^t if and only if -1 is not an eigen value [61]. Let $\lambda_n \leq \lambda_{n-1} \leq \dots \leq \lambda_2 < \lambda_1 = 1$ be the eigenvalues of A with $\lambda_1 = 1$. Also assume that $\gamma(A) = \max_{i>1} |\lambda_i|$. It can be shown that $\|J_i^{t+1} - \mathbf{J}_i^t\|^2 \leq \gamma^2 \|J_i^t - \mathbf{J}_i^t\|$. If $\gamma = 1$, then system fails to converge [61], [16].

Local Stochastic Gradient update is done as follows: $W_i^{t+1} = W_i^t - \eta_i^t s_i^t$ where $s_i^t = \frac{\partial J_i^t}{\partial W_i^t}(X_r, W_i^t)$, $X_r \in \mathbb{R}^{m_i}$ is the estimated gradient, W_i^t is the weight vector and η_i^t is the learning rate at node i at time t .

3.3 Convergence

The proof of convergence of the algorithm makes use of the following concept: In the distributed setting, the process of information exchange between k sites can be modeled as a non-stationary Markov chain. A non-stationary Markov chain is weakly ergodic if the dependence on the state distribution vanishes as time tends to infinity [60]. A detailed discussion regarding convergence of the algorithm is presented here.

First, we make the following assumptions about the cost function J :

Assumption 3:

1. There holds $J(W^t) \geq 0$, for every $W^t \in \mathbb{R}^m$
2. **Lipschitz Continuity of ∇J :** The function J is continuously differentiable and there exists a constant K_1 such that

$$\| \nabla J(W^{t_1}) - \nabla J(W^{t_2}) \| \leq K_1 \| W^{t_1} - W^{t_2} \|, \forall W^{t_1}, W^{t_2} \in \mathbb{R}^m. \quad (1)$$

3. If J satisfies the Lipschitz condition above, then

$$J(W^{t_1} + W^{t_2}) \leq J(W^{t_1}) + (W^{t_2})' \nabla J(W^{t_1}) + \frac{K}{2} \| W^{t_2} \|^2, \text{ for all } W^{t_1}, W^{t_2} \in \mathbb{R}^m.$$

In our algorithm, the vector W^t is split over sites S_1, S_2, \dots, S_k . The attributes at site $S_i, (1 \leq i \leq k)$ are updated according to the following equation:

$$W_i^{t+1} = W_i^t - \eta_i^t s_i^t \quad (2)$$

where η_i^t is the step size and s_i^t is the descent direction at site S_i . Let T^i be the set of times when processor i makes an update. It is assumed that $s_i^t = 0$ when $t \notin T^i$. For times $t \in T^i$, we assume that the update direction is such that the cost function decreases and s_i^t has the opposite sign from $\nabla J_i(W_i^t)$. The underlying deterministic gossip algorithm is described by:

$$W_i^{t+1} = \sum_{\{i|t \in T^i\}} \alpha_{ii} W_i^t + \sum_{\{j|t \in T^i\}} \alpha_{ij} W_j^t \quad (3)$$

where the coefficients α 's are non-negative scalars.

Example 1 Let S_i and S_j be the only two sites communicating with each other. Then equation 3 reduces to

$$\begin{aligned} W_i^{t+1} &= \alpha_{ii} W_i^t + \alpha_{ij} W_j^t \\ &= \alpha_{ii} (W_i^{t-1} - \eta_i^{t-1} s_i^{t-1}) + \alpha_{ij} (W_j^{t-1} - \eta_j^{t-1} s_j^{t-1}) \\ &= (\alpha_{ii} W_i^{t-1} + \alpha_{ij} W_j^{t-1}) - (\alpha_{ii} \eta_i^{t-1} s_i^{t-1} + \alpha_{ij} \eta_j^{t-1} s_j^{t-1}) \\ &= \dots \\ &= (\alpha_{ii} W_i^1 + \alpha_{ij} W_j^1) - (\alpha_{ii} (\eta_i^{t-1} s_i^{t-1} + \eta_i^{t-2} s_i^{t-2} \dots + \eta_i^1 s_i^1) \\ &\quad + \alpha_{ij} (\eta_j^{t-1} s_j^{t-1} + \eta_j^{t-2} s_j^{t-2} \dots + \eta_j^1 s_j^1)) \end{aligned}$$

Hence, by induction it can be shown that:

$$W_i^t = \sum_{j=1}^k \alpha_{ij} W_j^1 + \sum_{\tau=1}^{t-1} \sum_{j=1}^k \alpha_{ij} \eta_j^\tau s_j^\tau \quad (4)$$

It is also assumed that there exist positive constants K_4 and K_5 such that step-sizes η_i^t are bounded as follows: $\frac{K_4}{t} \leq \eta_i^t \leq \frac{K_5}{t}$. Furthermore, the following assumptions hold true:

Assumption 4:

1. $\forall i, j$ and $0 \leq \tau \leq t$, $0 \leq \alpha_{ij} \leq 1$.
2. For any i, j and $\tau \geq 0$, the limit of α_{ij} as t tends to infinity exists and is the same for all i and is denoted by α_i .
3. There exists some $\eta > 0$ such that $\alpha_j \geq \eta$ and $\forall j \in \{1, \dots, k\}$ and $\tau \geq 0$.
4. There exists constants $A > 0$ and $\rho \in (0, 1)$ such that $|\alpha_{ij} - \alpha_j| \leq A\rho^{t-\tau}, \forall t > \tau > 0$.

Assumption 5:

Descent Lemma [7] at each site:

- (a) For every i and t we have,

$$s_i^t \nabla J_i(W_i^t) \leq 0. \quad (5)$$

- (b) There exist positive constants K_2 and K_3 such that

$$K_2 |\nabla J_i(W_i^t)| \leq |s_i^t| \leq K_3 |\nabla J_i(W_i^t)| \quad (6)$$

Let $\mathcal{S}(t)$ be the set of random variables defined by: $\mathcal{S}(t) = \{s_i^\tau | i \in \{1, \dots, k\}, \tau < t\}$. The variables in $\mathcal{S}(t)$ are the only sources of randomness upto the time t at site i . The set $\mathcal{S}(t)$ is also a representation of the entire history of the algorithm upto the moment that the update directions s_i^τ are generated.

Assumption 6:

Stochastic Descent Lemma [7] at each site:

There exist positive constants K_6 , K_7 and K_8 such that:

(a)
$$\nabla J(W_i^t)' E[s_i^t | \mathcal{S}(t)] \leq -K_6 \|\nabla J(W_i^t)\|^2, \forall t \in T^i. \quad (7)$$

(b)
$$E[\|s_i^t\|^2 | \mathcal{S}(t)] \leq K_7 \|\nabla J(W_i^t)\|^2 + K_8, \forall t \in T^i. \quad (8)$$

Theorem 5 implies that the expected direction of the update given the past history is in the descent direction. In Theorem 6 the presence of constant K_8 in the inequality allows the algorithm to make non-zero updates even when the minimum has been reached.

Assumption 7:**Partial Asynchronism [7])**

There exists a positive integer B such that:

- (a) For every i and for every $t \geq 0$ at least one of the elements of the set $\{t, t+1, \dots, t+B-1\}$ belongs to T^i .
- (b) There holds $\max\{0, t-B+1\} \leq \tau_{ij}^t \leq t$, for all i and j and $t \geq 0$.

Finally, for completion we introduce the notions of *martingales* and the martingale convergence theorem(s) which are required for the proofs in this appendix.

A martingale is a model of a fair game where knowledge of past events never helps predict the mean of the future winnings. In general, a martingale is a stochastic process for which, at a particular time in the realized sequence, the expectation of the next value in the sequence is equal to the present observed value even given knowledge of all prior observed values at a current time⁵. A formal definition (using measure theory [58]) is given below:

Let (σ, \mathcal{F}, P) be a probability space. A martingale sequence of length n , is a sequence X_1, X_2, \dots, X_n of random variables and corresponding sub- σ fields $\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_n$ that satisfy the following relations:

- Each X_i is an integrable random variable which is measurable with respect to the corresponding σ -field \mathcal{F}_i .
- The sigma fields are increasing $\mathcal{F}_i \subset \mathcal{F}_{i+1}$ for every i
- For every $i \in [1, 2, \dots, n-1]$, we have the relation, $X_i = E[X_{i+1} | \mathcal{F}_i]$ almost everywhere P .

Along the same lines,

- A *submartingale* is defined as: for every i , $X_i \leq E[X_{i+1} | \mathcal{F}_i]$ almost everywhere P and
- A *supermartingale* is defined as: for every i , $X_i \geq E[X_{i+1} | \mathcal{F}_i]$ almost everywhere P .

Martingale convergence theorem is a special type of theorem since the convergence follows from the structural properties of the sequence of random variables. The Supermartingale Convergence theorem and a variant used in proofs is presented next.

Supermartingale Convergence Theorem [7]: Let $\{Y_i\}$ be a sequence of random variables and let $\{\mathcal{F}_i\}$ be a sequence of finite sets of random variables such that $\mathcal{F}_i \subset \mathcal{F}_{i+1}$ for each i . Suppose that:

- Each Y_i is non-negative
- For each i , we have $E[Y_i] < \infty$
- For each i , we have $E[Y_{i+1} | \mathcal{F}_i] \leq Y_i$ with probability 1.

Then there exists a non-negative random variable Y such that the sequence of $\{Y_i\}$ converges to Y with probability 1.

⁵ [http://en.wikipedia.org/wiki/Martingale_\(probability_theory\)](http://en.wikipedia.org/wiki/Martingale_(probability_theory))

An extension of the above theorem, can be stated as follows:

Let $\{Y_i\}$ and $\{Z_i\}$ be two sequence of random variables. Let $\{\mathcal{F}_i\}$ be a sequence of finite sets of random variables such that $F_i \subset F_{i+1}$ for each i . Suppose that:

- The random variables Y_i and Z_i are non-negative.
- There holds $E[Y_{i+1}|\mathcal{F}_i] \leq Y_i + Z_i, \forall i$ with probability 1.
- There holds $\sum_{i=1}^{\infty} E[Z_i] < \infty$.

Then there exists a nonnegative random variable Y such that the sequence $\{Y_i\}$ converges to Y with probability 1.

Proposition 1.0. Convergence of the DFE Algorithm Under the Assumptions 1.0-3.0, there exists some $\eta^0 > 0$, such that if $0 < \eta_i^t < \eta^0$, then:

1. $\lim_{t \rightarrow \infty} J(W_i^t)$ exists and is the same for all i with probability 1.
2. $\lim_{t \rightarrow \infty} (W_i^t - W_j^t) = 0$ with probability 1 and in the mean square sense.
3. For every i , $\lim_{t \rightarrow \infty} \nabla J(W_i^t) = 0$.
4. Suppose that the set $\{W | J(W) \leq C\}$ is bounded for every $C \in \mathcal{R}$; then there exists a unique vector W^* at which J is minimized and this is the unique vector at which ∇J vanishes. Then W_i^t converges to W^* for each i with probability 1.

Proof. Without loss of generality, assume that $\eta_i^t = \frac{1}{t}, \forall i, t$.

We note that the underlying gossip protocol illustrated by equation 3 has a simple structure but is not easy to manipulate in algorithms primarily because we have one such equation for each i and they are generally coupled. Thus we need to keep track of vectors $W_1^t, W_2^t, \dots, W_k^t$ simultaneously. Analysis would be simpler if we could associate one single vector \mathcal{W}^t that summarizes the information contained in W_i^t 's. Let \mathcal{W}^t be defined as follows:

$$\mathcal{W}^t = \sum_{i=1}^k \alpha_i W_i^1 + \sum_{\tau=1}^{t-1} \sum_{i=1}^k \alpha_i \eta_i^\tau s_i^\tau \quad (9)$$

The interpretation of vector \mathcal{W}^t is quite interesting in the following sense – if the sites stopped performing updates at time \bar{t} , but keep communicating and forming convex combinations of their states using the gossip protocol, they will asymptotically agree and the vector they agree upon is \mathcal{W}^t . Finally, $\mathcal{W}^{t+1} = \mathcal{W}^t + \sum_{i=1}^k \alpha_i \eta_i^t s_i^t$.

Define also the following:

$$b^t = \sum_{i=1}^k \|s_i^t\|, t \geq 1. \quad (10)$$

$$G^t = - \sum_{i=1}^k \nabla J(W_i^t)' \alpha_i s_i^t, t \geq 1. \quad (11)$$

Lemma 1.0 (a) If $t \in T^i$, then

$$E[G^t|S(t)] \geq K_6\lambda \sum_{\{i|t \in T^i\}} \|\nabla J(W_i^t)'\|^2 \geq 0, \quad (12)$$

where $\alpha_{ii} > \lambda, \forall i \in \{1, \dots, k\}$ and $\lambda > 0$.

(b) If $t \geq 1$, then

$$E[(b^t)^2|S(t)] \leq A_1 E[G^t|S(t)] + A_2 \quad (13)$$

where $A_1 = \frac{kK_7}{\lambda K_6}$ and $A_2 = k^2 K_8$

Proof: Using Assumption 2.0b, Equation 7 and the fact that $s^i = 0, t \notin T^i$, we have:

$$\begin{aligned} E[G^t|S(t)] &= - \sum_{i|t \in T^i} \nabla J(W_i^t) \alpha_i E[s_i^t|S(t)] \\ &\geq \sum_{i|t \in T^i} K_6 \|\nabla J(W_i^t)\|^2 \alpha_i \\ &\geq \lambda K_6 \sum_{i|t \in T^i} \|\nabla J(W_i^t)\|^2 \end{aligned}$$

This proves part (a) of the Lemma. Applying Equation 8 we obtain,

$$\begin{aligned} E[(b^t)^2|S(t)] &= E[(\sum_{i=1}^k \|s_i^t\|)^2|S(t)] \\ &\leq k \sum_{i=1}^k E[\|s_i^t\|^2|S(t)] \\ &\leq k \sum_{i=1}^k (K_7 \|\nabla J(W_i^t)\|^2 + K_8) \\ &\leq \frac{kK_7}{\lambda K_6} E[G^t|S(t)] + k^2 K_8 \end{aligned}$$

where the last inequality uses the proof in part (a). Q.E.D.

Lemma 2.0 For every $t \geq 1$, we have

$$\|\mathcal{W}^t - W_i^t\| \leq A \sum_{\tau=1}^{t-1} \frac{1}{\tau} \rho^{t-\tau} b^\tau \quad (14)$$

where $A > 0, \rho \in (0, 1)$

Proof: Subtracting Equation 9 from Equation 4 we have

$$\mathcal{W}^t - W_i^t = \sum_{\tau=1}^{t-1} \sum_{j=1}^k \frac{1}{\tau} [\alpha_j - \alpha_{ij}] s_j^t \quad (15)$$

Furthermore, using Assumption 1.1 (a) and definition 10 we obtain,

$$\begin{aligned}\|\mathcal{W}^t - W_i^t\| &\leq \sum_{\tau=1}^{t-1} \frac{1}{\tau} \sum_{j=1}^k A \rho^{t-\tau} \|s_j^\tau\| \\ &\leq A \sum_{\tau=1}^{t-1} \frac{1}{\tau} \rho^{t-\tau} b^\tau [Q.E.D]\end{aligned}$$

Using the fact that $\mathcal{W}^{t+1} = \mathcal{W}^t + \sum_{i=1}^k \alpha_i \eta_i^t s_i^t$ and Assumption 1.0 (3) we obtain:

$$\begin{aligned}J(\mathcal{W}^{t+1}) &= J(\mathcal{W}^t + \sum_{i=1}^k \alpha_i \eta_i^t s_i^t) \\ &\leq J(\mathcal{W}^t) + \sum_{i=1}^k \alpha_i \eta_i^t s_i^t \nabla J(\mathcal{W}^{t+1})' + \frac{K}{2} \left\| \sum_{i=1}^k \alpha_i \eta_i^t s_i^t \right\|_2^2 \\ &\leq J(\mathcal{W}^t) + \frac{1}{t} \sum_{i=1}^k \alpha_i s_i^t \nabla J(W_i^{t+1})' + \frac{1}{t} \sum_{i=1}^k \alpha_i s_i^t (\nabla J(\mathcal{W}^{t+1})' - \nabla J(W_i^{t+1})') + \frac{K}{2t^2} \|s_i^t\|_2^2 \\ &\leq J(\mathcal{W}^t) - \frac{1}{t} G(t) + \frac{1}{t} \sum_{i=1}^k \alpha_i s_i^t K_1 \|\mathcal{W}^{t+1} - W_i^{t+1}\| + \frac{K}{2t^2} (b^t)^2 \\ &\leq J(\mathcal{W}^t) - \frac{1}{t} G(t) + \frac{K_1}{t} \sum_{i=1}^k \alpha_i s_i^t A \sum_{\tau=1}^{t-1} \frac{1}{\tau} \rho^{t-\tau} b^\tau + \frac{K}{2t^2} (b^t)^2 \\ &\leq J(\mathcal{W}^t) - \frac{1}{t} G(t) + \frac{K_1 A}{t} b^t \sum_{\tau=1}^{t-1} \frac{1}{\tau} \rho^{t-\tau} b^\tau + \frac{K}{2t^2} (b^t)^2 \\ &\leq J(\mathcal{W}^t) - \frac{1}{t} G(t) + K_1 A \sum_{\tau=1}^{t-1} \frac{1}{t\tau} \rho^{t-\tau} b^\tau b^t + \frac{K}{2t^2} (b^t)^2 \\ &\leq J(\mathcal{W}^t) - \frac{1}{t} G(t) + K_1 A \sum_{\tau=1}^{t-1} \rho^{t-\tau} \left(\frac{(b^\tau)^2}{\tau^2} + \frac{(b^t)^2}{t^2} \right) + \frac{K}{2t^2} (b^t)^2 \\ &\leq J(\mathcal{W}^t) - \frac{1}{t} G(t) + A_3 \sum_{\tau=1}^t \rho^{t-\tau} \frac{(b^\tau)^2}{\tau^2}\end{aligned}\tag{16}$$

where $A_3 = \frac{K_1 A}{1-\rho} + \frac{K}{2}$

Lemma 3.0 There holds

$$\sum_{t=1}^{\infty} \frac{1}{t} E[G^t] < \infty\tag{17}$$

We take expectations of both sides of the above inequality and use Equation 13 to bound $E[(b^t)^2]$. This yields:

$$E[J(\mathcal{W}^{t+1})] \leq E[J(\mathcal{W}^t)] - \frac{1}{t}E[G^t] + A_3 \sum_{\tau=1}^t \rho^{t-\tau} \frac{1}{\tau^2} (A_1 E[G^\tau] + A_2). \quad (18)$$

Let $t = 1, 2, \dots, \bar{t}$ and add the resulting inequalities from Equation 18. Then,

$$\begin{aligned} E[J(\mathcal{W}^{\bar{t}+1})] &\leq J(\mathcal{W}^1) - \sum_{t=1}^{\bar{t}} \frac{1}{t} E[G^t] + A_2 A_3 \sum_{t=1}^{\bar{t}} \sum_{\tau=1}^t \rho^{t-\tau} \frac{1}{\tau^2} + A_1 A_3 \sum_{t=1}^{\bar{t}} \sum_{\tau=1}^t \rho^{t-\tau} \frac{1}{\tau^2} E[G^\tau] \\ &= J(\mathcal{W}^1) - \sum_{t=1}^{\bar{t}} \frac{1}{t} E[G^t] (1 - A_1 A_3 \frac{1}{t} \sum_{\tau=1}^{\bar{t}} \rho^{t-\tau}) + A_2 A_3 \sum_{t=1}^{\bar{t}} \sum_{\tau=1}^t \rho^{t-\tau} \frac{1}{\tau^2} \\ &\leq J(\mathcal{W}^1) - \sum_{t=1}^{\bar{t}} \frac{1}{t} E[G^t] (1 - \frac{A_1 A_3}{t(1-\rho)}) + A_2 A_3 \sum_{\tau=1}^{\bar{t}} \frac{1}{\tau^2 (1-\rho)} \end{aligned}$$

The term $A_2 A_3 \sum_{\tau=1}^{\bar{t}} \frac{1}{\tau^2 (1-\rho)}$ is bounded since the infinite sum $\sum_{\tau=1}^{\infty} \frac{1}{\tau^2 (1-\rho)}$ is bounded. If $\sum_{t=1}^{\infty} \frac{1}{t} E[G^t] = +\infty$, then the right hand side would equal $-\infty$. However, the left hand side is non-negative. This proves the lemma. [Q.E.D]

Lemma 4.0 The sequence $\{J(\mathcal{W}^t)\}$ converges with probability 1.

Proof: Taking conditional expectation of inequality 16, conditioned on $\mathcal{S}(t)$ and using Lemma 1.0 we have,

$$E[J(\mathcal{W}^{t+1})|\mathcal{S}(t)] \leq J(\mathcal{W}^t) + A_3 \sum_{\tau=1}^t \rho^{t-\tau} \frac{1}{\tau^2} E[(b^\tau)^2|\mathcal{S}(t)] \quad (19)$$

Let $Z(t) = \sum_{\tau=1}^t \rho^{t-\tau} \frac{1}{\tau^2} E[b^{\tau^2}|\mathcal{S}(t)]$. Using Lemma 1.0 (b) and Lemma 3.0 we have:

$$\begin{aligned} \sum_{t=1}^{\infty} E[Z(t)] &= \frac{1}{1-\rho} \sum_{t=1}^{\infty} \frac{1}{t^2} E[(b^\tau)^2] \\ &\leq \frac{1}{1-\rho} \sum_{t=1}^{\infty} \frac{1}{t^2} (A_1 E[G^t] + A_2) \\ &< \infty. \end{aligned} \quad (20)$$

Using inequalities 19 and 20, a variant of the Supermartingale theorem applies and hence $\{J(\mathcal{W}^t)\}$ converges with probability 1.[Q.E.D]

4 Empirical Evaluation

4.1 Aims

Our objective is to investigate empirically the utility of the consensus-based algorithm we have described. We use $Model(k, f)$ to denote the model returned by the consensus-based algorithm in Section 3.2 using k nodes in a network, each of which can call on an ILP engine to construct at most f features. In this section, we compare the performance of: $Model(N, F)$ ($N > 1$) with $Model(1, N \times F)$. The latter effectively represents the model constructed in a non-distributed manner, with all features present at a single centralised node. For simplicity, we will call the former the *Distributed* model and the latter the *Centralised* model.

We intend to examine if there is empirical support for the conjecture that the performance of the *Distributed* model is better than that of the *Centralised* model. We are assuming that the performance of a model-construction method is given by the pair (A, T) where A is an unbiased estimate of the predictive accuracy of the classifier, and T is an unbiased estimate of the time taken to construct a model. In all cases, the time taken to construct a model also includes the time taken to identify the set of features by the ILP engine and the time to compute their values. When $k > 1$, the time will also include time for exchanging information. Comparison of pairs (A_1, T_1) and (A_2, T_2) will simply be lexicographic comparisons.

4.2 Materials

Data Data for experiments are in two categories:

1. **Synthetic.** We use the “Trains” problem posed by R. Michalski for controlled experiments. Datasets of 1000 examples are obtained for randomly drawn target concepts (see “Methods” below).⁶ For this we use S.H. Muggleton’s random train generator⁷ that defines a random process for generating examples. We will use this data for controlled experiments to test principal conjecture about the comparative performances of *Distributed* and *Centralised* models.
2. **Real.** We report results from experiments conducted using some well-studied real world biochemical toxicology problems (Mutagenesis [31]; Carcinogenesis [32]; and DssTox [?]). Our purpose in examining performance on the real-data is twofold. First, we intend to see if the use of linear models is too restrictive for real problems. Second, we would like to see if the results obtained on synthetic data are reflected on real-world problems. We note that for these problems predictive accuracy is the primary concern.

⁶ We note here that we are not concerned with large numbers of examples here, since the main investigation is concerned with subsets of the feature-space, and not of the data instances.

⁷ <http://www.doc.ic.ac.uk/~shm/Software/GenerateTrains/>

Algorithms and Machines The DFE algorithm has been implemented on a Peer-to-Peer simulator, PeerSim [39]. This software sets up the network by initializing the nodes and the protocols to be used by them. The newscast protocol, an epidemic content distribution and topology management protocol is used. Nodes can perform actions on local data as well as communicate with each other by selecting a neighbour to communicate with (using an underlying overlay network). In each communication step, they mutually update their approximations of the value to be calculated, based on their previous approximations. The emergent topology from newscast protocol has a very low diameter and is very close to a random graph ([25],[26]).

The ILP system used in all experiments is Aleph [54]. The latest version of this program (Aleph 6) is available from the second author. The Prolog compiler used is Yap (version 6.2.0). The programs are executed on a dual Quad-Core AMD Opteron 2384 processors equipped with 2.7 GHz processors, 32 GB RAM, and local storage of 4×146 GB 15K RPM Serial attached SCSI (SAS) hard disks.

4.3 Method

For the synthetic data, we distinguish between “simple” targets (comprising disjuncts of 1-4 features) and “complex” targets (comprising disjuncts of 8–12 features).⁸ We call this dimension “Target”. Our method for experiments is straightforward:

1. For each value of *Target*
 - (a) Randomly draw a target concept from *Target*
 - (b) Classify each data instance as + or – using the target concept
 - (c) Randomly generate a network with N nodes
 - (d) For each node in the network:
 - i. Set the number of iterations T and initialize the learning parameter η_i for the node. It is assumed that all nodes agree on the initial choice of T and $\eta_i = \eta$.
 - ii. Execute the algorithm described in Section ?? for T iterations and the ILP engine restricted to constructing F features
 - iii. Record the predictive accuracy A of the (local) model along with the time T taken to construct the model (this includes the feature construction time, and the feature computation time). The pair (A, T) is the performance of the *Distributed* model for the concept.
 - (e) Using a network with a single node:
 - i. Execute the algorithm described in Section ?? for T iterations, learning parameter η , and the ILP engine restricted to constructing $N \times F$ features

⁸ This distinction between *simple* and *complex* is based on results from cognitive psychology which suggest that people find it difficult to remember concepts with larger than 7 disjuncts.

- ii. Record the predictive accuracy A' of the mode along with the time taken to construct the model T' (again, this includes the feature construction time and feature computation time). The pair (A', T') is the performance of the *Centralised* model for the concept.
2. Compare the performances of the *Distributed* and the *Centralised* models for the concepts.

The following additional details are relevant:

1. Two sources of sampling variation result with this method. First, variations are possible with the target drawn in Step 1a. Second, to ensure that both *Distributed* and *Centralised* approaches are constructing features from the same feature-space, we employ the facility within Aleph of drawing features from an explicitly defined feature space (this is specified using a large tabulation of features allowed by the language constraints). Although only “good” features are retained (see below), sampling variations can nevertheless result for both *Distributed* and *Centralised* models from step of drawing features. In effect, we are performing a randomised search for good features within a pre-defined feature space. We report averages for 5 repetitions of draws for the target, and 5 repetitions of the randomised search for a given target.
2. A target is generated as follows. For simple targets, the number of features is chosen randomly from the range 1 to 4. For complex concepts, the number of features is randomly chosen from the range 8 to 12. Features are then randomly constructed using the ILP engine, and their disjunction constitute the target concept.
3. As noted previously, data instances for controlled experiments are drawn from the “Trains” problem. The data generator uses S.H. Muggletons random train generator. This implements a random process in which each data instance generated contains the complete description of a data object (nominally, a “train”).
4. An initial set of parameters needs to be set for the ILP engine to describe “good” features. These include C , the maximum number of literals in any acceptable clause constructed by the ILP system; Nodes, the maximum number of nodes explored in any single search conducted by the ILP system; Minacc, the minimum accuracy required of any acceptable clause; and Minpos, the minimum number of positive examples to be entailed by any acceptable clause. C and Nodes are directly concerned with the search space explored by the ILP system. Minacc and Minpos are concerned with the quality of results returned (they are equivalent to “precision” and “support” used in the data mining literature). We set $C = 4$, Nodes=5000, Minacc=0.75 and Minpos=2 for our experiments here. There is no principled reason for these choices, other than that they have been shown to work well in the literature.
5. The parameters for the PeerSim simulator include the size of the network, degree distribution of the nodes and the protocol to be executed at each node. We report here on experiments with a distributed network with $N = 10$ nodes. Each of these nodes can construct up to $F = 500$ features (per class)

and the centralised approach can construct up to $N \times F = 5000$ features (per class).

6. The experiments here use the Hinge loss function. The results reported are for values of T that the stochastic gradient descent method starts to diverge.
7. The learning rate η_i remains a difficult parameter in any SGD-based method. There is no clear picture on how this should be set. We have adopted the following domain-driven approach. In general, lower values of the learning rate imply a longer search. We use three different learning rates corresponding to domains requiring high, moderate and low amounts of search (corresponding to complex, moderate or simple target concepts). The corresponding learning rates are 0.01, 0.1 and 1. We reiterate that there is no prescribed method for deciding these values, and better results may be possible with other values. The maximum number of iterations T is set to a high value (1000). The algorithm may terminate earlier, if there are no significant changes to its weight vector.
8. Since the tasks considered here are binary classification tasks, the performance of the ILP system in all experiments will be taken to be the classification accuracy of the model produced by the system. By this we mean the usual measure computed from a 2×2 cross-tabulation of actual and predicted classes of instances. We would like the final performance measure to be as unbiased as possible by the experimental estimates obtained during optimization, and estimates are reported on a holdout set.
9. With results from multiple repetitions (as we have here), it is possible to perform a Wilcoxon signed-rank test for both differences and accuracy and differences in time. This allows a quantitative assessment of difference in performance between the Distributed and the Centralised models. However, results with 5 repetitions are unreliable, and we prefer to report on a qualitative assessment, in terms of the average of accuracy and time taken.

A data instance in each of the real datasets is a molecule, and contains the complete description of the molecule. This includes: (a) bulk properties, like molecular weight, logP values etc.; and (b) the atomic structure of the molecule, along with the bonds between the atoms. For these datasets, clearly there are no concepts to be drawn, and sampling variation results solely from the feature-construction process. We therefore only report on experimental results obtained from repeating the randomised search for features. Again, estimates of predictive accuracy are obtained from a holdout set. For mutagenesis and carcinogenesis, each of the 10 computational nodes in the distributed network constructs up to 500 features, and the centralised approach constructs up to 5000 features (per class). For DssTox, we found there were fewer high precision features than the other two datasets. So the nodes in the distributed network constructs up to 50 features and the centralised node up to 500 features (per class)

4.4 Results

We present first the main results from the from the experiments on synthetic data (shown in Fig.3). The primary observations in these experiments are as follows:

(1) On average, as concepts vary, the distributed algorithm appears to achieve higher accuracies than the centralised approach, although the differences may not be significant for a randomly chosen concept; (2) On average, as concepts vary, the time taken for model construction by the distributed approach can be substantially lower⁹; and (3) The variation in both accuracies and time with the distributed approach due to both changes in the concept, or due to repetitions of feature-construction appear to be less than the centralised approach.

Taken together, these results suggest that good, stable models can be obtained from the distributed approach fairly quickly, and that the approach might present an efficient alternative to a centralised approach in which all features are constructed by a single computational unit.

Model	<i>Simple</i>		<i>Complex</i>	
	Acc.(%)	Time (s)	Acc. (%)	Time (s)
<i>Centr.</i>	83.3(14.4)	451.3(206.0)	92.0(7.1)	222.8(139.6)
<i>Distr.</i>	93.4(10.5)	54.4(15.3)	98.7(1.2)	41.6(13.7)

(a)

Model	<i>Simple</i>		<i>Complex</i>	
	Acc.(%)	Time (s)	Acc. (%)	Time (s)
<i>Centr.</i>	83.6(20.1)	107.7(50.2)	95.3(0.6)	410.4(8.8)
<i>Distr.</i>	79.9(10.4)	39.6(9.5)	96.6(0.6)	52.9(2.7)

(b)

Fig. 3. Results on synthetic data comparing *Centralised* and *Distributed* models. The results in (a) are averages from repetitions across concepts; and in (b) are averages from repetitions of the feature-construction process for a randomly drawn concept. In all cases, *Distributed* denotes the model obtained with a 10 node network, each of which employs a randomised search for up to 500 good features. *Centralised* denotes the model obtained with a single node employing a randomised search for up to 5000 good features. All randomised searches draw features from the same feature-space.

What can we expect from the consensus-based learner on the real datasets? Results are in Fig.4), and we observe the following: (1) There is a significant difference in accuracies between the distributed and centralised models on two

⁹ Although not apparent in the tabulation, the time is dominated by the time for constructing features. As a result, we note that the ratio of times for the centralised and distributed approaches need not be (linearly) proportional to the number of nodes in the network. For example, the search for a large subset of good features conducted by the a centralised approach may take much longer than the search for several small subsets conducted by the distributed approach.

of the datasets (*Canc330* and *DssTox*). On balance, we cannot conclude from this that there is either one of the models is better; (2) As with the synthetic data, the time for the distributed models is substantially lower. As before, the time is dominated by the feature-construction effort. For the real data sets, it appears that it is substantially easier to get smaller subsets of good features than larger ones (as observed from the substantial difference in the times between the distributed and centralised models); and (3) Comparisons against the baseline suggest that the use of linear models is not overly restrictive, since the models obtained are not substantially worse (predictively speaking) than the ones obtained by ILP in the past.

Again, taken together, these results provide support to the trends observed with the controlled experiments and suggest that the distributed approach would continue to perform at least as well as the centralised approach on real data.

Model	<i>Mut188</i>		<i>Canc330</i>		<i>DssTox</i>	
	Acc.(%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)
<i>Centr.</i>	84.3(2.6)	141.5(9.1)	67.6(0.5)	553.2(25.1)	53.8(0.0)	377.9(23.3)
<i>Distr.</i>	76.8(0.0)	2.1(0.2)	56.8(0.5)	34.9(5.7)	61.6(1.0)	26.9(1.5)
Baseline	84.6(2.6)	—	50.4(2.8)	—	64.7(2.0)	—

Fig. 4. Results on real data comparing *Centralised* and *Distributed* models. The Baseline models are the ones reported in [56] (these are cross-validation estimates, whereas the estimates for *Centralised* and *Distributed* models are from holdout sets). No estimates of time are reported in that paper.

We turn now to some issues that have been brought out by the experiments:

The learning rate λ . As will all methods based on stochastic gradient descent, the central parameter remains the learning rate λ . Many strategies have been suggested in literature to automatically adjust the learning rates. (see for example [8], [10], [17], [57]). In general, the learning rate on an iteration η_i of the algorithm here is of the form $\eta_i = \frac{B}{T-\alpha}$; $B = e^{-\lambda T}$; $0 < \alpha \leq 1$. For experiments reported above, we have used fixed values of λ based on our assessment of the search required (see the additional details in the Methods section). The choice of λ is dataset dependent and hence this parameter needs to assigned in some domain-dependent manner (as we have done here).

Convergence Accuracy. In experiments here (both with synthetic and real data), we have observed that the predictive accuracy of the model from the distributed setting is comparable to the predictive accuracy from the non-distributed setting. Unlike gains in time which can be expected from a distributed setting, it is not evident beforehand what can be expected on the accuracy front. This is because the models constructed in the two settings sample different sets of features. The results here suggest a conjecture that

the consensus-based approach will always converge to model that is within some small error bound of the model from a centralised approach with the same number of features. We have some reason to believe that this conjecture may hold in some circumstances, based on the use of Sanov’s theorem [50] and related techniques.

5 Conclusion

A particularly effective form of Inductive Logic Programming has been its use to construct new features that can be used to augment existing descriptors of a dataset. Experimental studies reported in the literature have repeatedly shown that the relational features constructed by an ILP engine can substantially assist in the analysis of data. Models constructed in this way have looked at both classification and regression, and improvements have resulted in each case. Practical difficulties have remained to be addressed though. The rich language of first-order logic used by ILP systems engenders a very large space of possible new features. The resulting computational difficulties of finding interesting features is not easily overcome by the usual ILP-based methods of language bias or constraints. In this paper, we have introduced what appears to be the first attempt at the use of a distributed algorithm for feature selection in ILP which also has some provable guarantees of convergence. The experimental results we have presented suggest that the algorithm is able to identify good models, using significantly lesser computational resources than that needed by a non-distributed approach.

There are a number of ways in which the work here could be extended further. Conceptually, we have outlined a conjecture in the previous section that we believe is worth investigating further. If it is proven to hold, then this would be a first-of-its-kind result for consensus-based methods. In implementation terms, we are able to extend the approach we have proposed to other kinds of models that use convex loss functions, and to consider a consensus-based version of the SNoW architecture. This latter will give us the ability to partition very large datasets, and to deal with very large feature-spaces at once. is also not required within the approach that all computational nodes draw from the same feature space (this was a constraint imposed here to evaluate the centralised and distributed models in a controlled manner). It may be both interesting and desirable for nodes to sample from different feature-spaces, or with different support and precision constraints. Finally, We need to investigate whether certain kinds of network topologies are better than others (currently, we impose no control, and use a randomly generated network). Experimentally, we recognise that results on more real-world datasets are always desirable: we hope the results here will provide the impetus to explore distributed feature construction by ILP on many more real datasets.

Acknowledgements

H.D. is also an adjunct assistant professor at the Department of Computer Science at IIIT, Delhi and an Affiliated Member of the Institute of Data Sciences, Columbia University, NY. A.S. also holds visiting positions at the School of CSE, University of New South Wales, Sydney; and at the Dept. of Computer Science, Oxford University, Oxford.

References

1. Agarwal, A., Chapelle, O., Dudík, M., Langford, J.: A reliable effective terascale linear learning system. *Journal of Machine Learning Research* **15**, 1111–1133 (2014). URL <http://jmlr.org/papers/v15/agarwal14a.html>
2. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE (1995)
3. Antunes, C., Oliveira, A.L.: Generalization of pattern-growth methods for sequential pattern mining with gap constraints. In: MLDM (2003)
4. Aseervatham, S., Osmani, A., Viennet, E.: bitspade: A lattice-based sequential pattern mining algorithm using bitmap representation. In: ICDM (2006)
5. Ayres, J., Gehrke, J., Yiu, T., Flannick, J.: Sequential pattern mining using a bitmap representation. In: KDD (2002)
6. Benezit, F., Dimakis, A., Thiran, P., Vetterli, M.: Order-optimal consensus through randomized path averaging. *Information Theory, IEEE Transactions on* **56**(10), 5150–5167 (2010)
7. Bertsekas, D.P., Tsitsiklis, J.N.: *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, MA. (1997)
8. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Y. Lechevallier, G. Saporta (eds.) *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pp. 177–187. Springer, Paris, France (2010). URL <http://leon.bottou.org/papers/bottou-2010>
9. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: *Optimization for Machine Learning*, pp. 351–368. MIT Press (2011). URL <http://leon.bottou.org/papers/bottou-bousquet-2011>
10. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: *Optimization for Machine Learning*, pp. 351–368. MIT Press (2011)
11. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. *IEEE/ACM Trans. Netw.* **14**(SI), 2508–2530 (2006)
12. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **3**(1), 1–122 (2011)
13. Carlson, A., Cumby, C., Rosen, J., Roth, D.: The snow learning architecture. Tech. Rep. UIUCDCS-R-99-2101, UIUC Computer Science Department (1999). URL <http://cogcomp.cs.illinois.edu/papers/CCRR99.pdf>
14. Chalamalla, A., Negi, S., Subramaniam, L.V., Ramakrishnan, G.: Identification of class specific discourse patterns. In: CIKM, pp. 1193–1202 (2008)
15. Christoudias, C., Urtasun, R., Darrell, T.: Unsupervised distributed feature selection for multi-view object recognition. Tech. Rep. MIT-CSAIL-TR-2008-009, MIT, CSAIL (2008)

16. Cybenko, G.: Dynamic load balancing for distributed memory multiprocessors. *Proceedings of the Journal of Parallel and Distributed Computing* **7**, pp. 279–301 (1989)
17. Darken, C., Moody, J.: Note on learning rate schedules for stochastic optimization. In: *Proceedings of the 1990 Conference on Advances in Neural Information Processing Systems 3, NIPS-3*, pp. 832–838 (1990)
18. Das, K., Bhaduri, K., Kargupta, H.: A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks. *Knowl. Inf. Syst.* **24**(3), 341–367 (2010)
19. Dekel, O., Gilad-Bachrach, R., Shamir, O., Xiao, L.: Optimal distributed online prediction using mini-batches. *J. Mach. Learn. Res.* **13**(1), 165–202 (2012)
20. Dimakis, A., Sarwate, A., Wainwright, M.: Geographic gossip: efficient aggregation for sensor networks. In: *Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on*, pp. 69–76 (2006)
21. Duchi, J., Agarwal, A., Wainwright, M.: Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling. *IEEE Transactions on Automatic Control* **57**(3), 592–606 (2012)
22. Garofalakis, M.N., Rastogi, R., Shim, K.: Spirit: Sequential pattern mining with regular expression constraints. In: *VLDB* (1999)
23. Han, Y., Wang, J.: An l1 regularization framework for optimal rule combination. In: *ECML/PKDD* (2009)
24. Jawanpuria, P., Nath, J.S., Ramakrishnan, G.: Efficient rule ensemble learning using hierarchical kernels. In: *ICML*, pp. 161–168 (2011)
25. Jelasity, M., Guerraoui, R., Kermarrec, A.M., Steen, M.: The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In: *Middleware 2004, Lecture Notes in Computer Science*, vol. 3231, pp. 79–98 (2004)
26. Jelasity, M., Montresor, A., Babaoglu, Ö.: Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.* **23**(3), 219–252 (2005)
27. Ji, X., Bailey, J., Dong, G.: Mining minimal distinguishing subsequence patterns with gap constraints. *Knowledge and Information Systems* (2006)
28. John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: *PROCEEDINGS OF THE ELEVENTH INTERNATIONAL Conference on Machine Learning*, pp. 121–129. Morgan Kaufmann (1994)
29. Joshi, S., Ramakrishnan, G., Srinivasan, A.: Feature construction using theory-guided sampling and randomised search. In: *ILP*, pp. 140–157 (2008)
30. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pp. 482–491 (2003)
31. King, R.D., Muggleton, S.H., Srinivasan, A., Sternberg, M.J.: Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences of the United States of America* **93**(1), 438–42 (1996)
32. King, R.D., Srinivasan, A.: Prediction of rodent carcinogenicity bioassays from molecular structure using inductive logic programming. *Environmental Health Perspectives* **104**, pp. 1031–1040 (1996). URL <http://www.jstor.org/stable/3433027>
33. Kudo, T., Maeda, E., Matsumoto, Y.: An application of boosting to graph classification. In: *NIPS* (2004)
34. Langford, J., Smola, A., Zinkevich, M.: Slow learners are fast. In: *Advances in Neural Information Processing Systems 22*, pp. 2331–2339 (2009)

35. Larson, J., Michalski, R.S.: Inductive inference of vl decision rules. *SIGART Bull.* (63), 38–44 (1977)
36. Liu, H., Motoda, H.: *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA (1998)
37. Mangasarian, L.: Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization* **33**(6), 1916–1925 (1995)
38. McDonald, R., Hall, K., Mann, G.: Distributed training strategies for the structured perceptron. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT '10*, pp. 456–464. Association for Computational Linguistics, Stroudsburg, PA, USA (2010)
39. Montresor, A., Jelasity, M.: PeerSim: A scalable P2P simulator. In: *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, pp. 99–100. Seattle, WA (2009)
40. Nagesh, A., Ramakrishnan, G., Chiticariu, L., Krishnamurthy, R., Dharkar, A., Bhattacharyya, P.: Towards efficient named-entity rule induction for customizability. In: *EMNLP-CoNLL*, pp. 128–138 (2012)
41. Nair, N., Saha, A., Ramakrishnan, G., Krishnaswamy, S.: Rule ensemble learning using hierarchical kernels in structured output spaces. In: *AAAI* (2012)
42. Niu, F., Recht, B., Ré, C., Wright, S.J.: Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems* **24**, 693–701 (2011)
43. Nowozin, S., Bakr, G., Tsuda, K.: Discriminative subsequence mining for action classification. In: *CVPR* (2007)
44. Pei, J.: Mining sequential patterns by pattern-growth: The prefixspan approach. *Journal of Machine Learning Research* **16-11** (2004)
45. Pei, J., Han, J., Wang, W.: Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems* (2005)
46. Pei, J., Han, J., Yan, X.F.: From sequential pattern mining to structured pattern mining: A pattern-growth approach. *Journal of Computer Science and Technology* **9**(3), 257–279 (2004)
47. Ramakrishnan, G., Joshi, S., Balakrishnan, S., Srinivasan, A.: Using ilp to construct features for information extraction from semi-structured text. In: *ILP*, pp. 211–224 (2007)
48. Roth, D.: Learning to resolve natural language ambiguities: A unified approach. In: *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, AAAI '98/IAAI '98*, pp. 806–813 (1998)
49. Saha, A., Srinivasan, A., Ramakrishnan, G.: What kinds of relational features are useful for statistical learning? In: *ILP* (2012)
50. Sanov, I.N.: On the probability of large deviations of random variables. In: *Mat. Sbornik*, vol. 42, pp. 11–44 (1957)
51. Shah, D.: Gossip algorithms. *Found. Trends Netw.* **3**(1), 1–125 (2009)
52. Specia, L., Srinivasan, A., Joshi, S., Ramakrishnan, G., Graas Volpe Nunes, M.: An investigation into feature construction to assist word sense disambiguation. *Machine Learning* **76**(1), 109–136 (2009)
53. Specia, L., Srinivasan, A., Ramakrishnan, G., das Graças Volpe Nunes, M.: Word sense disambiguation using inductive logic programming. In: *ILP*, pp. 409–423 (2006)
54. Srinivasan, A.: *The aleph manual* (1999). URL <http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/>

55. Srinivasan, A., Bain, M.: An empirical study of on-line models for relational data streams. Tech. Rep. 201401, School of Computer Science and Engineering, UNSW (2014)
56. Srinivasan, A., Ramakrishnan, G.: Parameter screening and optimisation for ilp using designed experiments. *Journal of Machine Learning Research* **12**, 627–662 (2011)
57. Sutton, R.: Adapting bias by gradient descent: An incremental version of delta-bar-delta. In: *In Proceeding of Tenth National Conference on Artificial Intelligence AAAI-92*, pp. 171–176. MIT Press
58. Tao, T.: *An introduction to measure theory* (2011)
59. Tsitsiklis, J.N.: *Problems in decentralized decision making and computation*. Ph.D. thesis, Department of EECS, MIT (1984)
60. Tsitsiklis, J.N., Bertsekas, D.P., Athans, M.: Distributed asynchronous deterministic and stochastic gradient optimization algorithms. In: *IEEE Transactions on Automatic Control*, vol. AC-31 (1986)
61. Varga, R.: *Matrix Iterative Analysis*. Prentice Hall, Englewood Cliffs, NJ (1962)
62. Zinkevich, M., Weimer, M., Smola, A.J., Li, L.: Parallelized stochastic gradient descent. In: *NIPS*, vol. 4, p. 4 (2010)